VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Záznamová kamera pro automobil

Dashboard camera

2015 Bc. Tomáš Růžička

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Tomáš Růžička**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Záznamová kamera pro automobil
Dashboard Camera

Zásady pro vypracování:

Cílem práce je vytvořit zařízení, které bude možno umístit do osobního či nákladního vozidla. Základem systému bude počítač Raspberry-Pi spolu s dalšími rozšiřujícími moduly. Jako moduly budou použity: GPS čidlo a akcelerometr. Volitelně bude použit i modul GSM/GPRS a modul pro sběrnici OBD2, případně modul displaye. Zařízení bude primárně sloužit jako "černá skříňka" vozidla. Bude ukládat a archivovat průběh cesty.

1. Proveďte průzkum trhu a zhodnoťte nedostatky obdobných výrobků.
2. Naprogramujte systém, který bude ukládat data z kamery, GPS a případně i dalších čidel.
3. Vytvořte vhodný systém pro prezentaci zaznamenaných dat.
4. Proveďte testování, zaměřte se především na stabilitu systému.
5. Zhodnoťte výsledky a porovnejte je s komerčními výrobky.

Seznam doporučené odborné literatury:

[1] RICHARDSON, Matt a Shawn P. WALLACE. Getting started with Raspberry Pi. 1. vyd. Sebastopol, California: O'Reilly Media, 2012, 161 s. ISBN 14-493-4421-6.
[2] UPTON, Eben. Raspberry Pi: uživatelská příručka. 1. vyd. Brno: Computer Press, 2013, 232 s. ISBN 978-80-251-4116-8.
[3] PILGRIM, Mark. Ponořme se do Python(u) 3: Dive into Python 3. Praha: Cz.Nic, 2010, 430 s. CZ.NIC. ISBN 978-80-904248-2-1.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. David Seidl, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015

.....

Na tomto místě bych rád poděkoval panu Ing. Davidu Seidlovi, Ph.D. za poskytnutí cenných rad a poznatků a také za trpělivost při vedení mé práce. Taktéž děkuji své rodině a přátelům za podporu.

Abstrakt

Růžička, T. *Záznamová kamera pro automobil*. Ostrava, 2015. Diplomová práce. Tato diplomová práce se zabývá problematikou open-source záznamových kamer do automobilů s využitím mikropočítače Raspberry Pi. První část práce je věnována zhodnocení aktuálního stavu trhu. Ve druhé části je vypracována implementace hardwarové a softwarové části systému. Softwarová část je naprogramována v jazyce Python s využitím také PyQt - do této části patří také operační systém Linux, kde bylo využito možností skriptovacího jazyka Bash.

Klíčová slova: Raspberry Pi, záznamová kamera, Linux, open-source, Python, PyQt, Bash

Abstract

Růžička, T. *Dashboard camera*. Ostrava, 2015. Diploma thesis. This diploma thesis deals with open-source dashboard cameras using microcomputers Raspberry Pi. The first part is focused on evaluation of the actual state of the problematics. The implementation of the hardware and software part of the whole system is mentioned in the second part of this thesis. The software part is programmed in Python also using PyQt - this part includes also the Linux operating system and scripting in Bash.

Keywords: Raspberry Pi, dashboard camera, Linux, open-source, Python, PyQt, Bash

Seznam použitých zkratek a symbolů

ANPR	–	Automatic Number Plate Recognition
ARM	–	Advanced RISC Machine
CSI	–	Camera Serial Interface
DMA	–	Direct Memory Access
DSI	–	Display Serial Interface
GPIO	–	General Purpose Input/Output
GPS	–	Global Positioning System
GUI	–	Graphical User Interface
HD	–	High Definition
HDMI	–	High-Definition Multimedia Interface
IR	–	Infra-Red
LED	–	Light-Emitting Diode
NMEA	–	National Marine Electronics Association
NOOBS	–	New Out Of the Box Software
OCR	–	Optical Character Recognition
PnP	–	Plug and Play
SD	–	Secure Digital
SDHC	–	Secure Digital High Capacity
SXGA	–	Super eXtended Graphics Array
USB	–	Universal Serial Bus
WDR	–	Wide Dynamic Range
WYSIWYG	–	What You See Is What You Get

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato diplomová práce pojednává o vytvoření open-source systému, který slouží k zaznamenávání průběhu cestování automobilem - čili lze na tento systém nahlížet jako na černou skříňku automobilu (na zařízení jako na černou skříňku se budu v této práci odkazovat často).

Stále častěji vídám na internetu video nahrávky řidičů, které zachycují nejen kuriózní, komické či tragické události při jízdě, ale také mnohdy pomáhají k dopadení těch, jenž svým chováním ohrožují ostatní. Tedy hlavní motivací a cílem této práce je nabídnout open-source řešení dostupné nejen profesionálním řidičům, ale řidičům obecně.

Na komerční produkty lze nahlížet jako na kompromis mezi cenou a deklarovanou funkcionalitou - avšak s využitím Raspberry Pi a různých zařízení se nabízejí další možnosti přidání vlastní funkcionality a s určitým vynaložením programátorského úsilí lze velmi jednoduše vytvořit vlastní, modifikovatelnou verzi černé skříňky.

Dovolím si rozdělit tuto práci na několik částí. První část bude zaměřena spíše na teoretickou stránku tématu. Píši zde o produktech, které je možno v dnešní době dostat na trhu - také zhodnocuji a porovnávám jejich funkcionalitu. V této části také zmiňuji obecné požadavky na záznamová zařízení pro automobily.

Samostatnou část vyhrazuji pro praktickou stránku této práce - implementaci. Na konci implementační části této práce uvádím poznatky z testování.

V poslední pasáži se věnuji zhodnocení výsledků a porovnání očekávaných výsledků s dosaženými.

2 Rešerše

Na trhu je dostupné velké množství záznamových zařízení pro automobily. Většinou se jedná o postačující standard, kde základem je kamera s mikrofonem a reproduktorem, slot pro microSD kartu - kde standardní podpora činí až 32 GB, USB konektor a dokonce malý displej. Propracovanější, ale pochopitelně dražší verze mají zabudovaný GPS čip, G-sensor pro zaznamenávání nárazu a kvalitnější displeje.

2.1 Hodnocení produktů

K porovnání jsem vybral několik produktů, které jsou dostupné na trhu - budu zohledňovat funkce, jenž nabízejí, kvalitu nahrávání ve dne a v noci, přítomnost různých čipů a konektorů a v neposlední řadě také cenu.

2.1.1 GoClever DVR LITE

Jedná se o nejlevnější kameru do automobilu na trhu. Zařízení disponuje 2.5" LCD displejem, vícejazyčným uživatelským rozhraním, vestavěným reproduktorem i mikrofonem. Umožňuje zaznamenat video s rozlišením až 1280x960 pixelů (kódování SXGA).

Zápory	- automatické mazání záznamů - krátké sekvence na nahrávání (maximálně 15 minut) - špatná kvalita nahrávání v noci - velká prodleva při automatické detekci pohybu - vícejazyčné rozhraní neobsahuje češtinu
Klady	+ cena (okolo 500,- Kč) + jednoduché ovládání + velikost zařízení

Tabulka 1: Kamera GoClever DVR LITE

2.1.2 iGET GUARD F80HD

Jak název napovídá, tato kamera disponuje záznamem s HD kvalitou (1280x720 pixelů nebo interpolované FullHD video). Rozlišení kamery je 5 MPix s infračerveným nočním přisvícením. V zařízení je integrovaný 2.7" LCD displej. Česká (i slovenská) lokalizace. MiniUSB spolu s vestavěným reproduktorem a mikrofonem jsou samozřejmostí.

Zápory	- deklarovaná kvalita záznamu neodpovídá skutečnosti - výdrž baterie - velká prodleva při automatické detekci pohybu - špatná kvalita nahrávání v noci (slabé IR diody) - nestabilní systém (úsporný režim se aktivuje náhodně, pomalé vyvažování barev a jasu, po naplnění paměťové karty dojde k jejímu smazání)
Klady	+ cena (okolo 900 Kč) + jednoduché ovládání + velikost zařízení

Tabulka 2: Kamera iGET GUARD F80HD

2.1.3 TrueCam A3

Kamera do automobilu s FullHD rozlišením a možností pořizovat 5Mpix fotografie. V zařízení je integrovaný G-sensor. Nechybí ani detekce pohybu a novinkou je funkce pro zamykání pro ochranu důležitých záznamů. Cena okolo 1.200,- Kč.

Zápory	- výdrž baterie - špatná kvalita nahrávání v noci - G-sensor - oslňující LED
Klady	+ jednoduché ovládání + velikost zařízení + velký displej + dlouhý napájecí kabel

Tabulka 3: Kamera TrueCam A3

2.1.4 CEL-TEC E07

Kvalitní palubní kamera s 2.7" barevným LCD displejem. Záznam videa ve FullHD kvalitě. Integrovaný G-senzor a funkcí pro lepší obraz v noci WDR. Cena okolo 2.500,- Kč.

Zápory	nemá zásadní nedostatky
Klady	+ kvalitní obraz (čitelné SPZ za šera i za tmy) + cena/výkon + kvalita nahrávání v noci + dlouhý napájecí kabel + jednoduché ovládání

Tabulka 4: Kamera CEL-TEC E07

2.1.5 TrueCam A5

Tato digitální videokamera do automobilu nabízí širokoúhlý objektiv, funkci automatického spuštění. Zařízení disponuje 2.7" LCD displejem a vícejazyčným uživatelským rozhraním, včetně češtiny. Integrovaný G-senzor má možnost nastavení citlivosti v rozmezí 2G až 8G. V tomto modelu je vestavěný GPS přijímač (spolupracuje s Google Maps). Možnost nahrávání ve FullHD 1920x1080 rozlišení (30 snímků za vteřinu). Zařízení má zabudované HDMI rozhraní. Cena okolo 3.300,- Kč.

Zápory	nemá zásadní nedostatky
Klady	+ kvalitní ostrý obraz (ve dne i v noci) + GPS + jednoduché přehledné ovládání + možnost nasazení polarizačních filtrů + podsvětlená tlačítka

Tabulka 5: Kamera TrueCam A5

2.1.6 Eltrinex LS400W

Toto zařízení vyniká vysokou kvalitou zaznamenaného videa ve FullHD rozlišení a technologií WDR (což umožňuje rozpoznávat registrační značky vozidel i osoby). Je zde možnost také uzamknout pořízené záznamy. Integrovaný senzor nárazu. Záznam videa probíhá ve smyčce, kdy jsou starší záznamy přemazávány novějšími - avšak důležité záznamy lze velmi jednoduše zamknout - zamknuté video nelze přepsat ve smyčce. Cena okolo 4.500,- Kč.

Zápory	- absence GPS - cena
Klady	+ velmi kvalitní ostrý obraz (ve dne i v noci) + záznam ve smyčce

Tabulka 6: Kamera Eltrinex LS400W

2.2 Porovnání návrhu mého řešení s komerčními produkty

Návrh řešení záznamové kamery chci prezentovat jako open-source projekt, který je v určitých směrech schopen konkurovat komerčním produktům. Tento projekt jsem nazval Raspberry Pi Dashcam - záznamová kamera do automobilu, která vyniká 5Mpix kamerou s FullHD kvalitou nahrávání až 30 snímků za sekundu. Má zabudovaný GPS modul, G-sensor a také existuje možnost připojení až 128GB microSD karty. Nechybí zde ani HDMI konektor a dva USB konektory. Cena HW se pohybuje okolo 3.400,- Kč.

Zápory	- cena - absence displeje a reproduktoru
Klady	+ kvalitní ostrý obraz + záznam ve smyčce + GPS, G-sensor + jednoduché ovládání

Tabulka 7: Kamera Raspberry Pi DashCam

3 Požadavky na systém, motivace

V předešlé části jsem se věnoval zhodnocení produktů a v této části chci navázat.

V prvé řadě jde o to, aby systém uměl data pořídit, zpracovat je a vhodně zobrazit. Základem systému pro sběr dat je mikropočítač Raspberry Pi spolu s danými moduly. Uživatel s tímto systémem bude interagovat prostřednictvím několika mikrospínačů. A protože motivací je vytvořit černou skříňku, tak v prvé řadě podotýkám, že na černé skříňce samotné nebudou přítomna žádná výstupní zařízení v podobě displeje či reproduktoru.

Černá skříňka bude schopná synchronizovat pořízená data s domácím úložištěm prostřednictvím domácí sítě.

Bude použito jednoduchého uživatelského rozhraní pro manipulaci s daty. Logika aplikace bude naprogramována v jazyce Python s využitím Bash skriptů. V aplikaci bude možno přehrávat pořízené videozáznamy a zobrazovat průběh trasy s vyhodnocením statistik.

Obrázek 1: Schématicky naznačená funkcionalita systému

4 Implementace

V této kapitole budou popsány dva pohledy na systém, a sice pohled hardwarový a softwarový.

4.1 Hardwarová část

V následující části popíši výběr HW, který jsem k vytvoření záznamové kamery použil a způsob připojení čidel a jiných komponent k Raspberry Pi.

4.1.1 Raspberry Pi model B revize 2

Základem černé skříňky je mikropočítač Raspberry Pi (obrázek 2), který je velký přibližně jako platební karta. Svou velikostí nikoho neohromí, je však velmi všestranný.

Základem tohoto počítače je jednočipový počítač BCM2835 firmy Broadcomm, který obsahuje procesor ARM1176JZF-S s taktem 700 MHz.

O vykreslování vizuálních dat se stará výkonný grafický procesor VideoCore 4. Tento podporuje Open GL ES 2.0, hardwarovou akceleraci OpenVG a přehrávání videa ve FullHD kvalitě 1080p (30 snímků za sekundu).

Použitý model B má 512 MB paměti s DMA infrastrukturou.

Operační systém je zaváděn z SD karty - ve většině případů se jedná o Raspbian, ale nevylučují se ani distribuce Arch Linux nebo distribuce Android a jiné.

Použitý model je osazen konektorem RJ45 (ethernetový adaptér BaseT 10/100) a dvěmi USB porty verze 2.0. Nechybí ani female HDMI port, konektor pro CSI Raspberry Pi HD kameru, DSI port či 3.5 mm výstupní audio port.

Zařízení je napájeno pěti volty stejnosměrného napětí pomocí microUSB konektoru (5 V, odběr je příjemných 700 mA).

Na základní desce mikropočítače je 26 GPIO pinů.

Obrázek 2: Mikropočítač Raspberry Pi model B revize 2

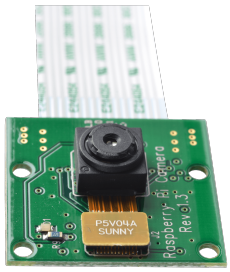
4.1.2 Kamera pro Raspberry Pi

Černá skříňka má jednu z hlavních funkcí nahrávání videa v průběhu cesty.

Obecně kamera pro Raspberry Pi může být využita k pořizování HD videa a fotografií.

Modul má 5 megapixelovou kameru (viz obrázek 3) s fixním zaostřováním, podporuje módy 1080p30, 720p60 a VGA90.

Připojuje se pomocí 15 cm dlouhého plochého kabelu do CSI portu na Raspberry Pi a je kompatibilní se všemi modely.



Obrázek 3: Modul kamery pro Raspberry Pi

4.1.3 GPS čidlo

Jedná se o modul Ultimate GPS ver. 3 od firmy Adafruit (obrázek 4).

Základem GPS modulu je čipset MTK3339 - GPS čidlo, které může spolupracovat až s 22 satelity na 66 kanálech s velmi dobrou citlivostí sledování trasy (-165 dB) a vestavěnou anténou.

Tento modul má velmi nízký odběr proudu a sice 20 až 25 mA (v aktivním stavu). Lze jej napájet 3.3 až 5 V.

Modul je mimo jiné opatřen diodou, která indikuje stav připojení (FIX), uFL (respektive IPEX) konektorem pro připojení externí antény a vestavěným záznamovým systémem. Za dobrých podmínek je GPS čidlo připraveno pracovat přibližně 30 sekund po zapnutí.

Obrázek 4: Adafruit GPS Ultimate verze 3.0

4.1.4 Akcelerometr

Použil jsem tří-osý akcelerometr ADXL345 (obrázek 5) s digitálním I^2C a SPI rozhraním.

Senzory v tomto zařízení mají tři osy měření (X, Y a Z) a lze nastavovat různé citlivosti a to buď $\pm 2g$, $\pm 4g$, $\pm 8g$ nebo $\pm 16g$ (nižší rozsah je lepší pro rozpoznávání pomalých pohybů, vyšší rozsah se používá pro vysokorychlostní sledování průběhu).

Obrázek 5: Akcelerometr ADXL345

4.1.5 Ostatní

Dále jsem použil tyto komponenty:

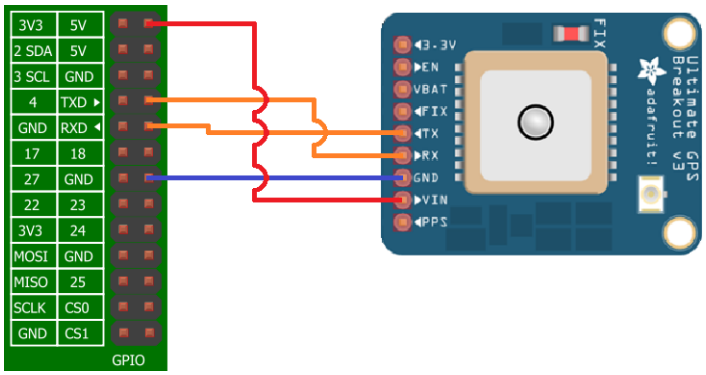
- nepájivé prototypové pole
- Pi T-Cobbler Breakout Kit
- sadu propojovacích kabelů
- 3 mikrospínače DTS 21R
- 3 rezistory $10k\Omega$
- 8GB SDHC paměťovou kartu Class 10
- externí GPS anténu s 3m kabelem
- USB PnP WiFi adaptér
- USB PnP mikrofon
- kabel USB2microUSB 2m (pro napájení Raspberry Pi)
- duální (1.1A a 2.0A, 12 až 18V) USB adaptér do zapalovače v automobilu

Pro zvýšení počtu USB portů lze použít aktivní USB hub (doporučuji 4-portový, spolu s jeho USB napájecím kabelem). Do USB portů lze připojit Wi-Fi adaptér, mikrofon, bluetooth adaptér a jiná zařízení (např. USB flash disk pro zvýšení kapacity místa pro nahrávání záznamu).

Volitelně lze použít bezdrátový ELM327 OBD2 modul, který se k Raspberry Pi připojuje pomocí Bluetooth - k Raspberry Pi musí samozřejmě být připojen USB PnP Bluetooth adaptér. Dále lze volitelně použít modul dotykového displeje SainSmart 3.2" TFT LCD s mezikusem pro připojení k Raspberry Pi - v tom případě je však potřeba použít další Raspberry Pi Cobbler a propojit příslušné piny prostřednictvím nepájivého pole.

4.1.6 Připojení čidel na GPIO a jiných komponent

Pro přehlednost jsem vytvořil sadu několika schémat a tabulek připojení modulů a dalších součástek k Raspberry Pi. Reálně jsou všechna čidla připojena na nepájivé pole a pomocí propojovacích drátů jsou propojeny příslušné piny. Celkově je pak nepájivé pole připojeno k Raspberry Pi pomocí rozšiřujícího zařízení Pi T-Cobbler Breakout Kit - toto však není vyžadováno a je možné pomocí propojovacích drátů typu female-female připojit čidla přímo k Raspberry Pi GPIO.



Obrázek 6: Připojení GPS na GPIO sběrnici

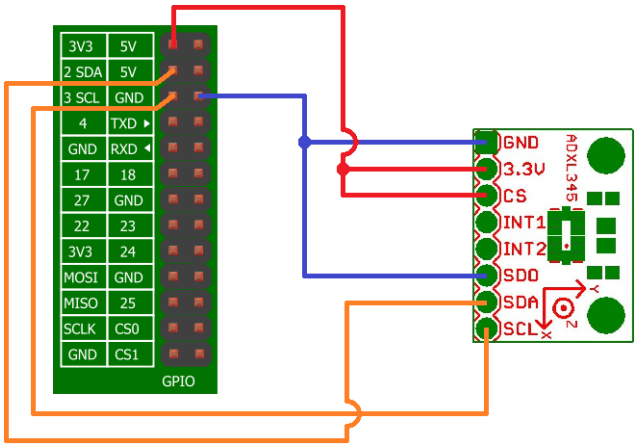
Na obrázku 6 je vyobrazené připojení čidla GPS na GPIO sběrnici. Pro větší přehlednost zapojení GPS čidla uvedl do tabulky 8.

Piny	
Raspberry Pi	**Adafruit GPS**
5V	VIN
GND	GND
RX	TX
TX	RX

Tabulka 8: Propojení dvojic pinů na Raspberry Pi a GPS čidle

Konkrétně toto GPS čidlo má odběr 20 mA, což je velmi nízká hodnota. K GPS modulu lze pomocí uFl2RP-SMA adaptéru připojit externí anténu pomocí SMA RG174 koaxiálního kabelu s přijímačem.

Zapojení akcelerometru je schématicky zachyceno na obrázku 7



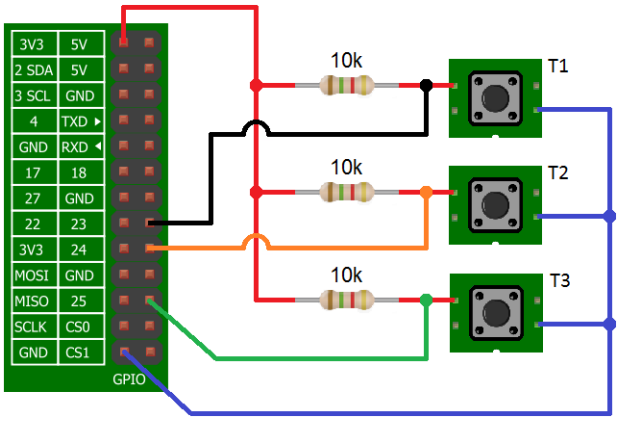
Obrázek 7: Připojení akcelerometru ADXL345 na GPIO sběrnici

Přehledně jsem zapojení akcelerometru uvedl do tabulky 9.

Piny	
Raspberry Pi	**ADXL345**
GND	GND
3V3	3V3
3V3	CS
GND	SD0
SDA	SDA
SCL	SCL

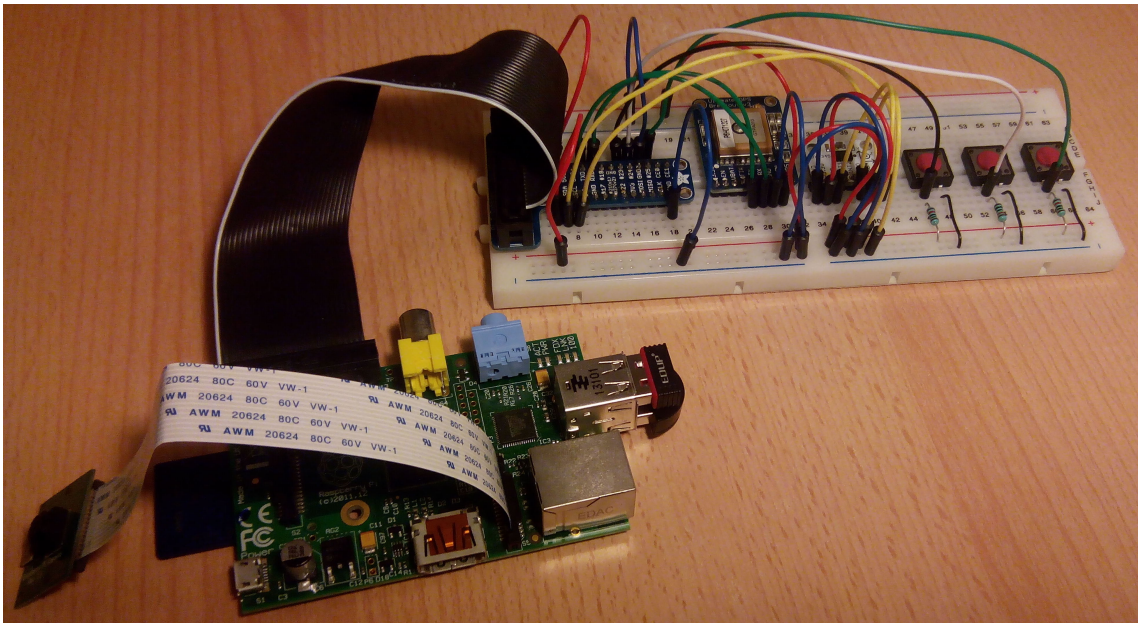
Tabulka 9: Propojení dvojic pinů na Raspberry Pi akcelerometru ADXL345

Schéma zapojení tlačítek je uvedeno na obrázku 8. Tlačítka jsou připojena k GPIO sběrnici na piny 23, 24 a 25.



Obrázek 8: Připojení mikrospínačů na GPIO sběrnici

Na obrázku 9 je možno vidět reálné zapojení celé černé skříňky.



Obrázek 9: Reálné zapojení černé skříňky

4.2 Softwarová část

Tato sekce obsahuje popis části systému, jenž manipuluje s daty z čidel. Samotnou manipulaci, zpracování a zobrazování dat řídí sada modulů (potažmo skriptů), napsaných v programovacím jazyce Python, spolu s využitím PyQt - PyQt je multiplatformní vazba mezi Qt a jazykem Python, používaná pro programování grafického uživatelského rozhraní. Dále jsem naprogramoval několik skriptů v programovacím jazyce Bash a neoddělitelnou součástí SW jsou konfigurační soubory.

Skripty jsou poměrně malé bloky kódu, kde klíčovou vlastností je produktivita, schopnost rychle měnit a testovat algoritmy a také velmi rychle dostat výsledky těchto testů. Skripty nemusí být vysoce efektivní, co se týče rychlosti zpracování, ale pokud jsou vhodně napsány, pak by se doba běhu skriptů neměla významně prodlužovat.

Obecně lze tento systém rozdělit na dva podsystémy - $raspberry$ a $desktop$. Oba podsystémy jsou od sebe odděleny a nesdílí spolu funkcionalitu - pouze si umí předávat data. Podsystém $raspberry$ je v roli sběrače dat a nemá GUI. Hlavní funkcí je nahrávání videa a volitelné nahrávání zvuku a zároveň pořizování dat z čidel GPS a akcelerometru.

Podsystém $desktop$ pracuje s pořízenými daty. Je zde obsažena veškerá logika manipulace s daty a jejich zobrazování - proto má tento podsystém oproti $raspberry$ implementované GUI, což uživateli umožňuje zpracovaná data prohlížet a analyzovat.

Mezi oběma systémy může probíhat komunikace typu klient-server. Do role serveru se staví podsystém $desktop$ - na pozadí běží služba, v žargonu správce unixových systému označována jako démon, která má za úkol přijímat data od klienta. V roli klienta je podsystém $raspberry$.

V softwarové části je popsána také funkcionalita mikrospínačů. Struktura systému je popsána následujícím schématem.

```
dashcam
├── system
│   ├── deskop
│   │   ├── configs
│   │   │   └── general_settings.conf
│   │   ├── modules
│   │   │   ├── simple_config_parser.py
│   │   │   ├── sync_data_server.py
│   │   │   ├── track_data_manager.py
│   │   │   └── vlc.py
│   │   ├── scripts
│   │   │   └── syncdataserverd
│   │   └── install.sh
│   └── raspberry
│       ├── configs
│       │   ├── audio_video.conf
│       │   └── general_settings.conf
│       ├── data
│       ├── modules
│       │   ├── adxl345.py
│       │   ├── button_manager.py
│       │   ├── gps_manager.py
│       │   ├── simple_config_parser.py
│       │   ├── syncdata_client.py
│       │   └── tracker.py
│       ├── scripts
│       │   ├── restart_gps
│       │   ├── trackd
│       │   └── video_audio
│       └── install.sh
├── data
├── gui
└── main.py
```

4.2.1 Konfigurační soubory

Zde bych se chtěl zmínit o konfiguračních souborech a práci s nimi. Některé konfigurační soubory jsou společné pro Python a Bash skripty, jiné jsou využívány buď Python nebo Bash skripty.

V jazyce Bash existuje velmi jednoduchá konstrukce, jak načítat data z konfiguračních souborů. Tato konstrukce vypadá následovně

source /cesta/ke/konfiguracnimu/souboru.conf

Je však nutno dodržet vnitřní strukturu souboru

promenna=hodnota

Pro programovací jazyk Python existuje knihovna, která dokáže číst konfigurační soubory a i v tomto případě je nutné dodržet vnitřní strukturu sdružování jednotlivých položek konfiguračního souboru do sekcí:

$[Sekce]$
promenna=hodnota

A jak jsem naznačil na začátku této kapitoly 4.2, s některými konfiguračními soubory se pracuje v obou jazycích. Naprogramoval jsem tedy jednoduchý wrapper Python knihovny ConfigParser, zvaný $simple\_config\_parser$, který umí vracet hodnoty proměnných bez použití sekcí, jsou-li jednotlivé položky souboru ve formátu $promenna = hodnota$.

4.2.2 Funkcionalita mikrospínačů

Funkcionalita hardwarových mikrospínačů (respektive tlačítek) je přehledně zachycena v tabulce 10, kde je možné vidět reakce systému na různé typy manipulace s těmito tlačítky.

Tlačítka	**Typ manipulace**	
	Krátké stisknutí	**Stisknutí delší než 5 sekund**
Tlačítko 1	Začátek/konec pořizování dat	Započetí synchronizace
Tlačítko 2	Restartování démona GPS	Restartování démona trackd
Tlačítko 3	Vypnutí Raspberry Pi (shutdown)	Restartování Raspberry Pi (reboot)

Tabulka 10: Funkcionalita mikrospínačů

4.2.3 Ukládání a zobrazování dat

4.2.4 Ukládání dat

Sběr dat probíhá ve smyčkách a je korigován démonem $trackd$, který spouští příslušný Python skript pro ukládání dat z GPS a akcelerometru. V tomto skriptu je dále aktivován skript pro nahrávání videa a volitelně také nahrávání zvuku - lze nastavit v konfiguračním souboru $audio\_video.conf$. Délka časové smyčky je taktéž uvedena v konfiguračním souboru $general\_settings.conf$. Ukládání započne (a také skončí) tehdy, když je krátce zmáčknuto tlačítko 1 (viz obrázek 8).

Výše zmíněný démon je v podstatě linuxový inicializační skript, kterému je nastaveno spouštění po startu operačního systému.

GPS čidlo je k Raspberry Pi připojeno přes sériové rozhraní a automaticky se připojuje do $/dev/ttyAMA0$. K zaznamenávání dat z GPS jsem použil knihovnu python-gps, která dokáže zpracovávat surová data ve formátu NMEA 0183 ze sériové linky.

Akcelerometr jsem k Raspberry Pi připojil na I^2C sběrnici - výchozí adresa tohoto zařízení je $0x53$. Naprogramoval jsem Python modul, který získává data z akcelerometru a na základě konfigurace vrací požadované informace z čidla.

Data z GPS (ale také z akcelerometru) jsou ukládána v předem specifikované časové periodě (jenž je uvedena v konfiguračním souboru *gps.conf*) ve formě středníkem oddělených hodnot do formátu $.trackdata$. V průběhu záznamu je kontrolováno, zda došlo k významnému zrychlení (respektive zpomalení) nebo k velkému náklonu ve směru některé ze tří os (například pokud bylo vozidlo převráceno či podobně) - v takovém případě je vytvořen tzv. zámek smyčky.

Zámkem smyčky je myšleno vytvoření souboru s příponou $.lock$ - název je shodný s názvem právě zaznamenávané smyčky. Tento soubor je prázdný. Pokud tedy dojde při nahrávání k potřebě vymazat nejstarší soubor pro uvolnění místa na disku, kontroluje se, je-li nejstarší smyčka zamčená - pokud ano, kontroluje se druhá nejstarší smyčka (respektive v pořadí druhá nejméně stará smyčka) a tak dále. Je také ošetřena situace, kdy jsou všechny smyčky zamknuté a je potřeba mazat soubory - v tom případě systém ukončí záznam trasy.

Nahrávání videa a zvuku probíhá paralelně s ukládáním dat z ostatních čidel. Video, které je pořizováno Raspberry Pi kamerou je komprimováno do standardního formátu H.264. Při výchozí konfiguraci se nahrává video s rozlišením 720 x 720 pixelů a 20 snímků za sekundu s trváním pět minut. Pro samotné nahrávání jsem použil program *raspivid*. Příkaz s výchozím nastavením vypadá takto

raspivid -w 720 -h 720 -n -hf -vf -fps 20 -o /cesta/video.h264 -t 300000

Zvuk je pořizován prostřednictvím PnP USB mikrofonu ve formátu ogg. Pro záznam zvuku je použitý program $arecord$, který z předem specifikovaného zařízení posílá surová data programu $oggenc$, jenž je do formátu ogg.

arecord -d 300 -D plughw:1,0 -f cd -t raw | oggenc - -r -o /cesta/audio.ogg

Název všech souborů formátován takto *„den_mesic_rok-hodina_minuta_sekunda“*.

Logika pořizování dat je navržena tak, aby při odpojení černé skříňky od zdroje napětí nedošlo ke ztrátě dat. Serializace dat na disk probíhá na konci každé iterace získávání dat z GPS a akcelerometru a to pomocí Python knihovny *pickle*. Pro uložení objektu do souboru je použití následující

pickle.dump(data, open(soubor, 'wb'))

V souboru $general\_settings.conf$ je nakonfigurována maximální kvóta využití místa na disku pro záznam dat. V průběhu záznamu je kontrolováno nepřekročení této kvóty - pokud se při nahrávání bude míra zaplnění blížit této kvótě, pak dojde ke smazání nejstaršího video záznamu (spolu s příslušným zvukovým záznamem) z důvodu uvolnění místa na disku. Avšak data z GPS a akcelerometru smazána nebudou a mohou být použita k pozdější analýze průběhu cesty.

4.2.5 Synchronizace dat

Na začátku této podkapitoly 4.2 jsem se zmínil o tom, že samotné Raspberry Pi bude sloužit pouze jako černá skříňka - čili jen pro záznam dat. Aby bylo možno data zobrazovat, je nutné, aby byla samotná data přesunuta do úložiště podsystému $desktop$, kde s nimi bude dále manipulováno.

Samotný přesun dat je realizován socketovou komunikací typu klient-server.

V roli serveru je počítač připojený do domácí sítě. Na tomto počítači běží socketový server, který poslouchá na předem definovaném portu a na předem definované IP adrese. Tento server je opět realizován formou inicializačního linuxového skriptu (démona).

Klient zná IP adresu serveru a také port, na kterém běží server. Na tuto adresu se na základě požadavku dotazuje, zda je možno zaslat data - požadavek pro zaslání dat generuje klient ($raspberry$) po podržení tlačítka 1 (viz schéma na obrázku 8). Připojení klienta do sítě lze realizovat jak připojením kabelu, tak bezdrátově za využití Wi-Fi adaptéru. Pokud server potvrdí svou dostupnost, pak dojde k navázání socketové komunikace a v daný okamžik se smí k serveru připojit právě jedna černá skříňka. Démon, jenž obsluhuje synchronizaci dat na straně klienta se v prvé řadě prokáže serveru svým sériovým číslem, které je uvedeno v souboru $/proc/cpuinfo$ - jedná se o sériové číslo procesoru černé skříňky - na tuto akci server zareaguje tak, že pokud není vytvořen adresář s názvem daného sériového čísla, dojde k vytvoření takového adresáře. Tento způsob byl zvolen z důvodu synchronizace několika černých skříněk s jedním serverem.

Dále je na straně klienta provedena archivace obsahu adresáře $data$ do formátu tar a započato samotné odesílání. Pokud nedojde během procesu odesílání k přerušení spojení či nenastane jiná chyba (nedostatek místa na úložišti serveru, apod.), pak je obsah adresáře $data$ vymazán.

```
def sendTAR():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((parserGeneral.getoption('ip'), int(parserGeneral.getoption('port'))))

    s.send("ID=%s" % get_serial_id())

    if create_tar():
        print "Start sending TAR file %s..." % tarfilename
        bufsize = 4096
            f = open(save_data_path + '/' + tarfilename, 'rb')
            l = f.read(bufsize)
            while (l):
                s.send(l)
                l = f.read(bufsize)
            f.close()

            print "Done sending TAR..."
            delete_tar(save_data_path + "/" + tarfilename)
            s.close()
            return True
     else:
            print "There is nothing to sent ... "
            s.close()
            return False
```

Výpis 1: Odesílání souboru na server

Pro lepší komunikaci HW a síťových zařízení je doporučeno použít hodnotu velikosti bufferu na relativně malou mocninu čísla 2 (já jsem použil 4096, jak je patrné ve výpisu kódu 1).

Démon, který obsluhuje serverovou část procesu synchronizace, čeká na připojení klienta a v okamžiku, kdy se daný klient připojí vytvoří v požadovaném adresáři nový tar soubor, do kterého zapíše všechna data ze socketového proudu - název tohoto adresáře a souboru je odvozen od data synchronizace. Poté je tento tar soubor rozbalen a vymazán z úložiště. V ukázce kódu 2 je zobrazeno přijímání dat serverem.

V tento okamžik jsou v úložišti serveru všechna data, nasbíraná černou skříňkou ve formátech $trackdata$, $h264$ a ogg.

```
while True:
    while True:
        (conn, addr) = s.accept() # establish connection with client
        print 'Got connection from', addr
        print "Start receiving data..."

        raspi_id = conn.recv(bufsize)
        if raspi_id.split('=')[0] == "ID":
            rpid_dirrname = "rpi_" + raspi_id.split('=')[1]
            rpid_dirpath = save_data_path + "/" + rpid_dirname
            assure_path_exists(rpid_dirpath)
            filename = get_filename() + ".tar"
            tf = open(rpid_dirpath + "/" + filename, 'wb')
            d = conn.recv(bufsize)
            while (d):
                tf.write(d)
                d = conn.recv(bufsize)
            tf.close()
            data = conn.recv(bufsize)
            if not data:
                untar(rpid_dirpath, filename)
                delete_tar(rpid_dirpath, filename)
                print "Done recieving data..."
                break
        else:
            print "Have not recieved the ID ..."
```

Výpis 2: Přijímání souboru

4.2.6 Analýza a zobrazování dat

Jakmile jsou všechna data uložena na serveru, může s nimi být dále manipulováno. Myšlenkou je využít skriptovacích možností jazyka Python a vytvořit automatický proces, jenž analyzuje a vizualizuje data.

Záleží na uživateli nad jakými daty chce provádět analýzu. Čili v první fázi procesu analýzy jsou vybrány $trackdata$ soubory. Vybrané soubory (jejich libovolný počet) je možno shlukovat do jednoho souboru a analýzu provádět nad tímto nebo lze analyzovat data jednotlivých smyček. Při shlukování záleží na pořadí jednotlivých vstupních souborů, proto musí být prvně seřazeny podle data jejich vytvoření.

```
for f in files_list :
    f = os.path.basename(f)
    filename = f.split('.')[0];
    extension = f.split('.')[1]

    temp = pickle.load(open(save_data_path + "/" + filename + "." + extension, 'rb')) #
        deserialize

    for i in temp:
        if "interval" in i:
            interval = i
            pass
        elif "speed_units" in i:
            speed_units = i
            pass
        else:
            lines.append(i)

lines.insert(0, speed_units)
lines.insert(0, interval)

pickle.dump(lines, open(save_data_path + "/" + get_merged_filename() + "_merged.trackdata",
    'wb'))
```

Výpis 3: Spojování několika trackdata souborů

Ve druhé fázi dochází k předzpracování všech dat. V první části předzpracování jsou vytvořeny seznamy, které budou uchovávat informace o zeměpisné šířce či délce, nadmořské výšce, rychlosti, času, počtu satelitů a o informacích z akcelerometru (záznamy ze všech tří os). Ve druhé fázi se provede deserializace záznamů ze souboru a tím se vytvoří seznam, jehož záznamy jsou ve formátu řetězců. Poté je každý řetězec zpracován a jeho obsah je rozdělen do výše zmíněných seznamů.

Analýza dat je prováděna především proto, aby data mohla být dobře vizualizována. Při analýze jsem narazil na několik problémů. Prvním z nich je fakt, že data z GPS jsou ve formátu zeměpisné šířky a zeměpisné délky. Z matematického hlediska lze na tyto údaje nahlížet jako na sférické souřadnice (tyto souřadnice leží na povrchu koule). Avšak pro vizualizaci sférických souřadnic do map či grafů je potřeba mít Kartézské souřadnice

bodů. Pro převedení sférických souřadnic do Kartézské soustavy jsem použil následujících dvou vzorců

$$x = zemepisna\_delka * NMI * 60 * \cos(zemepisna\_sirka)$$

$$y = zemepisna\_sirka * NMI * 60, NMI = 1852$$

Konstanta NMI je nastavena na hodnotu 1852, což je rovno jedné minutě na rovníku. Jedná se o velmi jednoduchou metodu, která poskytuje přesné výsledky v případě, že vzdálenosti mezi jednotlivými body jsou malé v poměru k poloměru zeměkoule. Vysvětlení je poměrně jednoduché - pokud se postupuje směrem k severnímu pólu, délka rovnoběžek se bude zkracovat (při dosažení severního pólu je délka rovnoběžky rovna nule). Takže při zeměpisné šířce $0°$, s každým stupněm zeměpisné délky je na rovníku uražená vzdálenost větší, než například na $45°$ zeměpisné délky. Proto je souřadnice x závislá jak na zeměpisné šířce, tak na zeměpisné délce a platí, že čím větší bude hodnota zeměpisné šířky, tím menší bude změna zeměpisné délky z hlediska vzdálenosti. Na druhou stranu souřadnice y není závislá na zeměpisné délce.

Planetu Zemi si lze představit jako kouli. Při vykreslování grafů (či map) se v podstatě provádí projekce povrchu Země na plochu.

Na obrázku 10 je zobrazena trasa, kterou jsem zakreslil do roviny pomocí Python knihovny $matplotlib$. Modrou barvou je vyznačena samotná trasa jízdy, což navíc indikuje, že se vozidlo pohybovalo. Navíc jsou na trase vyznačeny zelené, červené a černé body. Zelené body znamenají, že rychlost pohybu vozidla nepřevýšila předem daný práh, který je ve výchozím stavu nastaven na jeden km/h. Červené body indikují překročení rychlosti padesát km/h bez ohledu na to, prochází-li trasa obcí. Podobný význam mají černé body, jenž na obrázku 10 nejsou zaneseny - tyto indikují překročení rychlosti devadesát km/h. Dále je pomocí šipek zvýrazněn směr pohybu. Šipky jsou vykreslovány v periodě desetinásobku intervalu, ve kterém byl záznam pořízen (v tomto případě je interval roven jedné sekundě). Rotace každé šipky je v Pythonu počítána takto

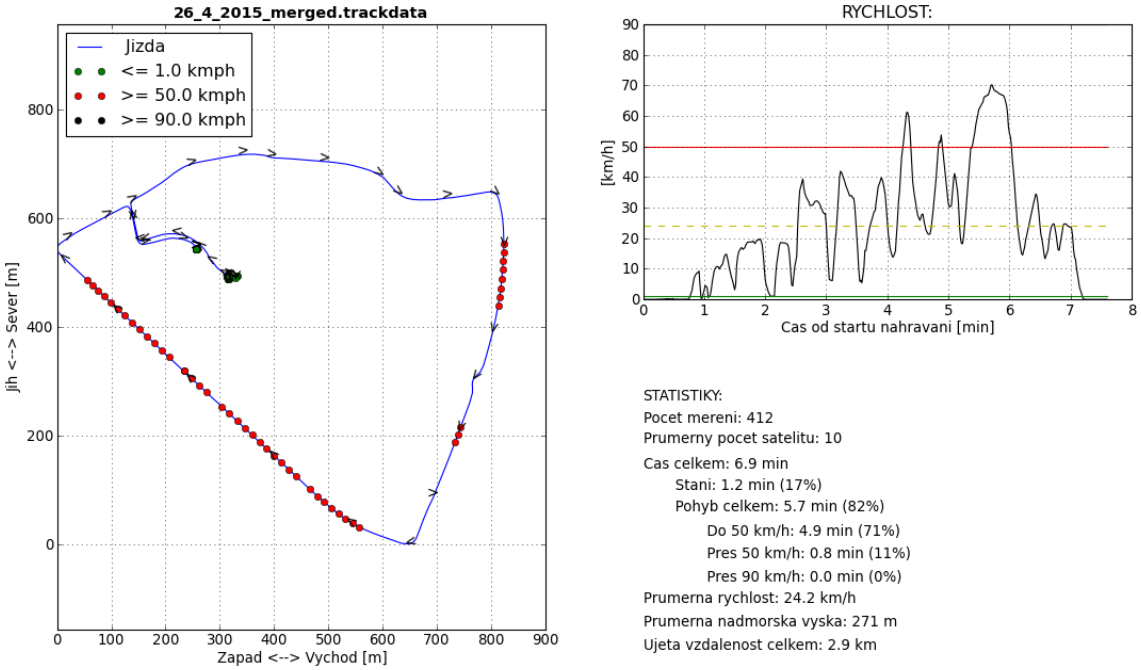
$$rotace = \frac{arctan2(y[i+1] - y[i], x[i+1] - x[i])}{rad}, rad = \frac{\pi}{180}$$

kde funkce $arctan2$ vrací hodnotu v radiánech a použitím konstanty rad je vrácen výsledek ve stupních. Výsledek tohoto výpočtu se dosadí do vykreslovací funkce z knihovny $matplotlib$.

V pravém horním rohu je vykreslen graf rychlosti s nastavenými prahy. Stejně jako při vykreslování jsou i zde rychlostní prahy barevně označeny. Graf vykresluje hodnoty aktuální rychlosti v čase - všechny tyto údaje jsou získány z GPS. Čas je zanesen na horizontální osu v jednotkách minut.

Do pravého dolního rohu jsem umístil přehledně uspořádané statistiky celé jízdy. Do těchto statistik jsem zahrnul celkový počet měření a průměrné hodnoty počtu satelitů, rychlosti a nadmořské výšky. Dále jsem celou trasu rozdělil do několika částí týkajících se časových úseků - zde jsem zadefinoval trvání jízdy a stání. Trvání jízdy jsem dále

rozdělil do částí „Do 50 km/h“, „Přes 50 km/h“ a „Přes 90 km/h“ - všechny údaje jsou uvedeny v minutách a jsou i percentuálně přepočteny.



Obrázek 10: Statistiky jízdy, které byly vygenerovány z celé dávky - tzn. z jedné synchronizace

Na tomto místě poznamenám, že ihned po synchronizaci dochází ke konverzi video souborů z formátu h264 do mp4. Zvolil jsem kontejner mp4, protože může obsahovat titulky a zvukové stopy. Samotná konverze je prováděna programem $MP4Box$.

MP4Box -add video.h264 video.mp4

Dále jsem použil program *ffmpeg*, který sloučí obrazovou a zvukovou stopu do jednoho *mp4* souboru - samozřejmě za předpokladu, že tyto existují. Následuje ukázka příkazu, který provede fúzi audio a video nahrávek

ffmpeg -i video.mp4 -i audio.ogg -map 0:1 -map 1:0 -vcodec copy -acodec copy video.mp4

Aby byly GPS souřadnice viditelné v přehrávaném videu, rozhodl jsem se tato data, která jsou v souborech $trackdata$, překonvertovat do formátu SubRip - tento formát používá příponu srt. Proto jsem v Pythonu naprogramoval funkci, která ze souboru $trackdata$ vezme požadovaná data a vytvoří SubRip rámce v závislosti na časování. V konfiguračním souboru $general\_config.conf$ na straně klienta jde totiž nastavit, s jakou periodou budou data z GPS (a akcelerometru) získávána a podle toho je nastaveno časování samotných titulků (výchozí nastavení je získávat data každou sekundu).

Jak již bylo zmíněno v úvodní kapitole, desktopová část systému má implementované uživatelské rozhraní. Toto rozhraní jsem navrhl v programu Qt Creator 4. Jedná se o jednoduchý WYSIWYG editor, který umožňuje GUI rychle a jednoduše modelovat. Výstupem tohoto programu jsou soubory ve formátu *ui* a použití těchto souborů v programovacím jazyce Python je dvojí. První možnost je použití utility *pyuic4*, která provede kompilaci *ui* souboru a vytvoří tak kód jazyka Python - příkaz vypadá následovně

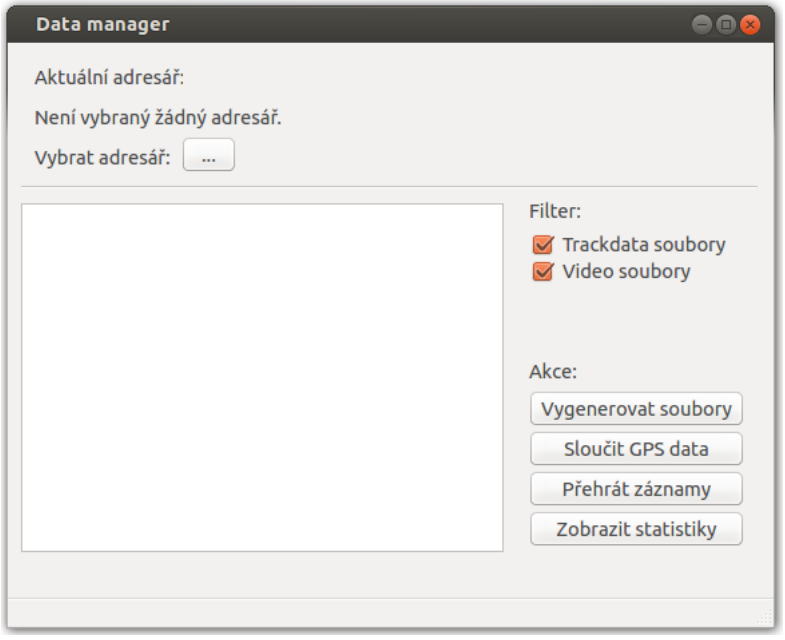
pyuic4 gui.ui -o gui.py

Já jsem využil možnosti druhé, a sice modulu *uic*, jehož použití je ukázáno ve výpisu kódu 4. Toto je mnohem jednodušší, než první možnost, protože odpadá nutnost spouštět další skript, který by kompiloval *ui* soubory. Modul *uic* to umí sám.

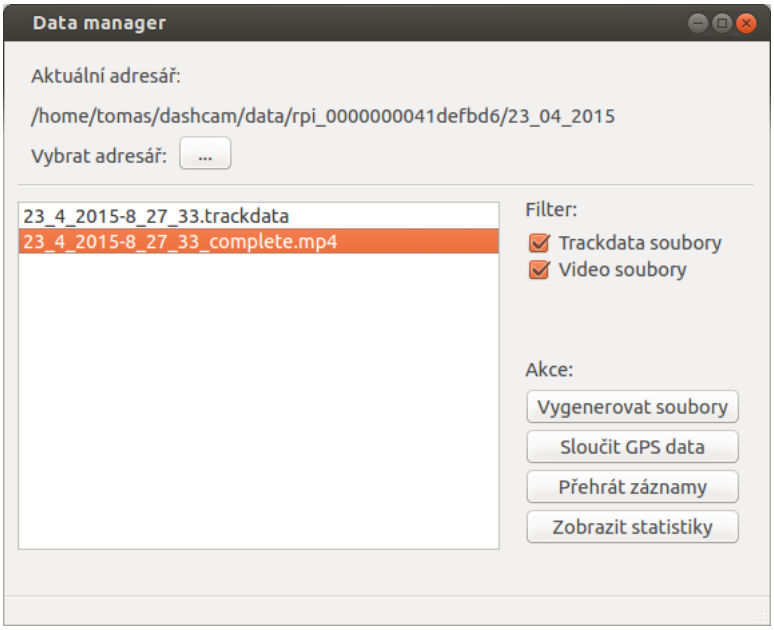
```
from PyQt4 import uic
...
gui = uic.loadUiType("gui/gui.ui")[0]
...
```

Výpis 4: Nahrání .ui souboru pomocí uic modulu

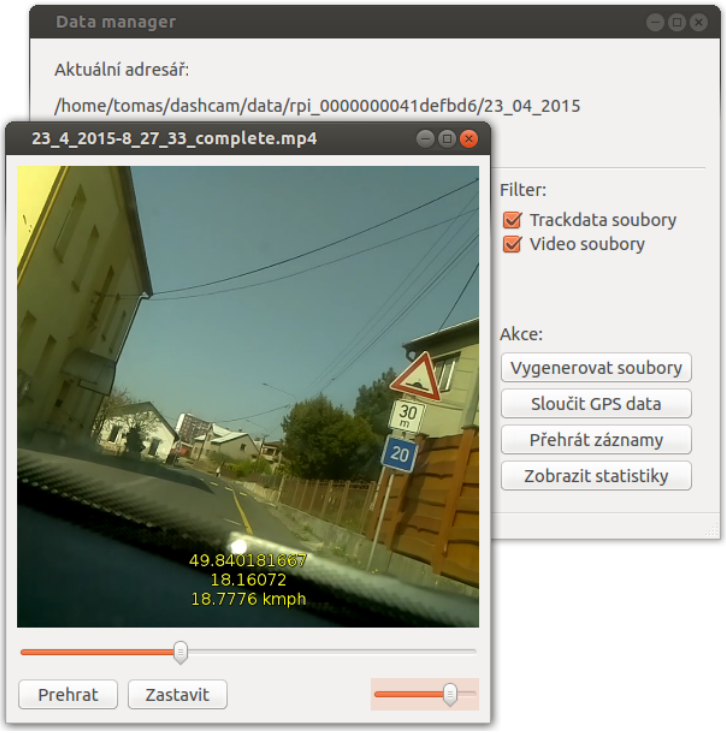
Na obrázcích 11, 12 a 13 je možno vidět ukázky uživatelského rozhraní jednotlivých oken.



Obrázek 11: Úvodní obrazovka desktopové aplikace



Obrázek 12: Obrazovka desktopové aplikace po vybrání adresáře s daty



Obrázek 13: Obrazovka desktopové aplikace při přehrávání vybraného souboru

V uživatelském rozhraní na obrázku 12 svou velikostí dominuje seznam souborů, se kterými je možno manipulovat - soubory, uložené ve vybraném adresáři lze filtrovat podle jejich typu (vyhledávají se soubory s příponami *mp4* a *trackdata*). Vybrané soubory *trackdata* lze po stisknutí tlačítka „Sloučit GPS data“ seskupovat do jednoho souboru a dále pro tento soubor zobrazovat statistiky. Jak jsem zmiňoval v podkapitole 4.2.6, ihned po synchronizaci dochází k vygenerování video souborů se zvukem a k překladu souborů *trackdata* do formátu *srt*. Avšak systém umí vyvolat tuto akci po stisknutí tlačítka „Vygenerovat soubory“.

Na obrázku 13 je možné vidět přehrávač *VLC*, který jsem integroval do PyQT uživatelského rozhraní - použil jsem externí Python knihovnu *vlc*. V rozhraní je možné ovládat přehrávání videa a volit hlasitost zvuku.

4.3 Možná rozšíření

V úvodní kapitole 1 jsem se zmiňoval o možnostech rozšíření tohoto systému. Hned v úvodní části této kapitoly prohlásím, že možnosti jsou takřka neomezené - co se týče HW i SW.

4.3.1 Rozpoznávání registračních značek vozidel

Automatické rozpoznávání registračních značek vozidel (ANPR), je prvním z možných rozšíření. Jedná se o metodu, která využívá optického rozpoznávání znaků v obrázcích (označováno jako OCR). Tato metoda byla vyvinuta v 80. letech ve Spojeném království v jednom z policejních výzkumných ústavů. Kroky algoritmu jsou tyto:

1. Lokalizace značky - nalezení a izolaci oblasti, kde se nachází registrační značka
2. Úprava orientace a velikosti značky
3. Normalizace - úprava jasu a kontrastu obrázku
4. Segmentace znaků
5. OCR
6. Syntaktická/geometrická analýza - porovnání pozic znaků vůči specifickým pravidlům různých států
7. Opakování výše uvedených kroků a průměrování hodnot pro dosažení lepších výsledků

Obrázek 14: ANPR

Existuje však několik obtíží, se kterými se musí SW umět vypořádat. Jedná se hlavně o nekvalitní vstupy a problém může nastat, když

- je značka na obrázku velmi malá a byla použita kamera s nízkým rozlišením
- vstupní obrázek byl rozmazaný (většinou vlivem pohybu vozidla)
- je špatné osvětlení objektů na obrázku a obrázek celkově má nízký kontrast díky přeexponování, odrazům či stínům

- je registrační značka zakryta jiným objektem (tažným zařízením) nebo když je značka samotná zaprášená

S některými výše uvedenými problémy je možné se vypořádat softwarově, avšak primárně jde o hardwarovou část.

4.3.2 Využití OBD2 sběrnice vozidla

Získávání informací z OBD2 sběrnice vozidla je dalším možným rozšířením tohoto systému (jak HW, tak i SW). Z OBD2 sběrnice lze získat proud dat přímo z řídící jednotky vozidla. Je potřeba použít diagnostický modul OBD2 - buď variantu s Bluetooth nebo s klasickým USB kabelem. OBD2 konektor má většina vozidel od roku výroby 1996, avšak ne každý diagnostický modul funguje v daném vozidle - vyzkoušel jsem levný Bluetooth diagnostický modul ELM327 (obrázek 15) ve voze VW Golf IV, vyrobeném v roce 1998 a nepodařilo se mi dostat vůbec žádná data z řídící jednotky. Poté jsem vyzkoušel ten samý diagnostický modul ve voze Škoda Octavia (který byl vyroben v roce 2002) a zde se mi podařilo stav řídící jednotky bez obtíží načíst - z této zkušenosti plyne nestabilita použití tohoto modulu.

Obrázek 15: Bluetooth diagnostický modul ELM327

Použití je velmi jednoduché. Stačí lokalizovat OBD2 rozhraní ve vozidle a připojit diagnostický modul. Poté stačí využít programovacího jazyka Python s knihovnou *pyserial*. Python kód 5 ukazuje, jak pomocí protokolu OBD2 získat aktuální rychlost.

```
import serial

def get_speed():
    s = serial.Serial("nazev_zarizeni", 38400, timeout=1)
    s.write("010D \r")
    speed_hex=s.readline().split(' ')
    speed_kmph=float(int('0x'+speed_hex[3], 0))

print "Speed" + get_speed() + "km/h"
```

Výpis 5: Získání rychlosti z OBD2

4.3.3 Použití modulu displeje

Dále je možné rozšířit systém o LCD dotkový 3.2"displej (obrázek 16) - bylo by však potřeba nahrát upravené jádro systému (tzv. kernel).

Obrázek 16: 3.2"LCD TFT displej

4.4 Instalace a konfigurace systému

V této kapitole jsem popsal proces instalace operačního systému Raspbian na Raspberry Pi a systému dashcam.

4.4.1 Instalace operačního systému na Raspberry Pi

Existují tři způsoby jak nainstalovat Raspbian. První z nich je stáhnout si ZIP archiv s instalačním manažerem NOOBS pro Raspberry Pi. Obsah tohoto archivu rozbalit na naformátovanou SD kartu (formát karty fat32). Poté stačí kartu vložit do Raspberry Pi a připojit jej ke zdroji napětí. Po zavedení instalačního manažera vybrat operační systém Raspbian a dokončit instalaci.

Druhá možnost je stáhnout *img* obraz systému Raspbian a za použití linuxové utility *dd* obsah obrazu rozbalit na SD kartu. To jde provést příkazem

sudo dd if=raspbian.img of=/dev/sdX bs=1M,

kterému se specifikuje cesta k *img* souboru a zařízení (*sdX*). Zařízení není potřeba formátovat, protože *img* již obsahuje souborový systém. Pak jako v předchozím případě stačí SD kartu vložit do Raspberry Pi a připojit jej ke zdroji. Zvolil jsem tuto možnost.

A nakonec poslední možnost je zakoupit si SD kartu s předinstalovaným systémem Raspbian.

4.4.2 Instalace a konfigurace systému dashcam na Raspberry Pi

Jako první krok je potřeba Raspberry Pi připojit do internetu. To lze realizovat buď připojením kabelu (není potřeba konfigurovat) nebo využít USB Wi-Fi adaptér.

1. sudo nano /etc/network/interfaces
   auto wlan0
   iface wlan0 inet dhcp
   wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

2. sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
   network={
   ssid="nazev_site"
   psk="heslo"
   proto=RSN
   key_mgmt=WPA-PSK
   pairwise=TKIP
   auth_alg=OPEN
   }

3. sudo reboot

Následně je nutno se k Raspberry Pi připojit pomocí SSH a nakopírovat zdrojové a konfigurační soubory do adresáře */home/pi/dashcam* a spustit instalační skript *sudo /home/pi/dashcam/install.sh*. Ten zajistí nainstalování všech potřebných balíků a nakopíruje potřebné inicializační skripty do adresáře */etc/init.d*. Pro další postup je nutno mít správně připojeny všechny potřebné moduly.

Poté je možno pokračovat v konfiguraci systému dashcam. Prvně je potřeba správně nakonfigurovat komunikaci s GPS čidlem.

1. sudo nano /boot/cmdline.txt
   změnit *dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait*
   na *dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait*

2. sudo nano /etc/inittab
   změnit *T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100*
   na *#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100*

3. sudo reboot

4. otestovat

   - sudo killall gpsd
   - sudo gpsd /dev/ttyAMA0 -F /var/run/gpsd.sock

- cgps -s ... měla by se objevit tabulka s údaji z GPS čidla

Dále je potřeba nakonfigurovat Raspberry Pi, aby komunikovalo se zařízeními, která jsou připojená na I^2C sběrnici - v tomto případě je to akcelerometr.

1. sudo nano /etc/modules
   přidat následující řádky:
   i2c-bcm2708
   i2c-dev

2. sudo nano /etc/modprobe.d/raspi-blacklist.conf
   změnit *blacklist i2c-bcm2708*
   na *#blacklist i2c-bcm2708*

3. sudo reboot

4. otestovat

   - sudo i2cdetect -y 1 ... zařízení by mělo být viditelné na adrese 53

Následující příkazy nakonfigurují funkcionalitu USB mikrofonu.

1. sudo modprobe snd_bcm2835

2. sudo nano /etc/modprobe.d/alsa-base.conf
   změnit *#options snd-usb-audio index=-2*
   na *options snd-usb-audio index=1*

3. sudo nano /etc/asound.conf
   přidat:
   pcm.!default {
   type plug
   slave {
   pcm "hw:1,0"
   }
   }
   ctl.!default {
   type hw
   card 0
   }

4. sudo reboot

5. otestovat

   - arecord -D plughw:1,0 -f cd -d 10 test_audio.wav

A nakonec zůstává pouze konfigurace Raspberry Pi kamery - to se provede následovně.

1. sudo raspi-config
   pokud v seznamu není možnost „camera“, pak je nutno ukončit raspi-config a spustit následující příkazy
   - sudo apt-get update
   - sudo apt-get upgrade
   - sudo raspi-config (znovu)
2. zvolit možnost „Camera“ a zvolit „Enable“ a potvrdit
3. zvolit „Finish“
4. sudo reboot
5. otestovat
   - test img: raspistill -o /home/pi/image.jpg ... kamera by měla vyfotit obrázek a uložit jej do adresáře */home/pi*

4.4.3 Instalace systému dashcam na počítači

Na počítači stačí pouze do domovského adresáře umístit celý adresář *dashcam* a spustit instalační skript */dashcam/scripts/install.sh* - tímto příkazem se opět doinstalují potřebné balíky.

Poté provést úpravu názvu domovského adresáře v souborech *track_data_manager.py* a *syncdata_server.py*

Uživatelské rozhraní systému se pak spouští příkazem

python /dashcam/main.py

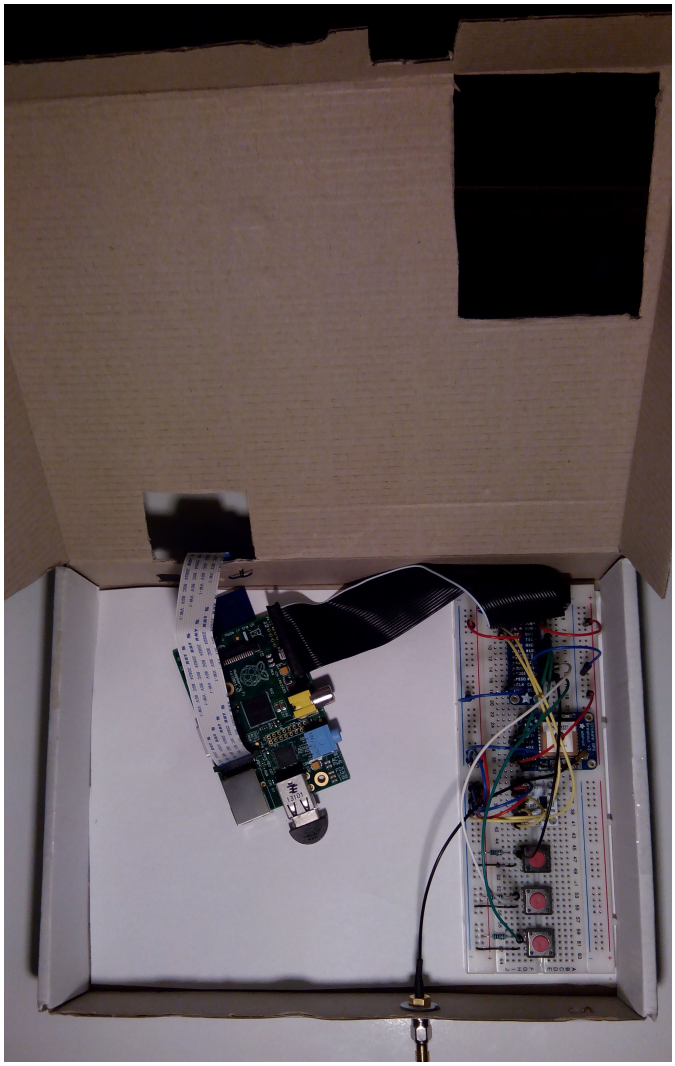
4.5 Testování systému

Na závěr jsem provedl testování celého systému a absolvoval také několik testovacích jízd.

Nejprve bych zde uvedl veškerá nastavení systému pro sběr dat. Na Raspberry Pi jsem nainstaloval operační systém Raspbian. V první fázi jsem testoval průběh instalace data sběrné části systému na Raspberry Pi. Po připojení mikropočítače do sítě jsem vzdáleně přes SSH spustil instalační skript. Všechny balíky byly nainstalovány bez problému - nyní přicházela na řadu konfigurace. Postupoval jsem krok za krokem tak, jak je uvedeno v předchozí kapitole a postupně se mi dařilo získávat testovací data z připojených čidel.

Druhá fáze testování systému se týkala instalace desktopové části. Opět jsem spustil instalační skript - tentokrát na svém notebooku, který byl připojený do domácí sítě prostřednictvím Wi-Fi. Instalace i konfigurace proběhly bez komplikací.

V tento okamžik byly všechny části systému nainstalovány a nakonfigurovány a konečně jsem mohl systém otestovat v reálných podmínkách. V podkapitole 4.1.6 jsem uvedl obrázek, kde bylo možno vidět prototyp HW části systému - vše zapojeno prozatímně do nepájivého pole. Celá tato sestava působila velmi subtilně a bylo potřeba vymyslet způsob, jak černou skříňku udržet na palubní desce a zabránit zklouzávání a eventuálně poškození HW. Proto jsem přišel na velmi jednoduché řešení - vsazení sestavy do papírové krabice. Do této krabice jsem vyřezal potřebné otvory a pevně na ni osadil kameru. Ukázku je možno vidět na obrázku 17



Obrázek 17: Uložení Raspberry Pi do krabice

Ponechal jsem všechna výchozí nastavení data sběrného systému a následně jsem prototyp černé skříňky nainstaloval před čelní sklo na palubní desku vozidla. Ve vozidle jsem připojil černou skříňku k napájecímu zdroji 5V/1A, který se již v minulosti

osvědčil - napájecí proud 1A je plně dostačující. V tomto případě se jednalo o napájecí USB adaptér, který se připojuje do zapalovače ve vozidle. Provedl jsem měření odběru elektrického proudu, který jsem zanesl do tabulky 11 a v porovnání s katalogovými hodnotami usuzuji, že celkový odhad odběru je uspokojivý (jen poznamenám, že připojení akcelerometru měřící přístroj nezaznamenal). Po připojení černé skříňky ke zdroji došlo k zavedení operačního systému z SD karty, které trvalo přibližně 20 sekund. Při testování jsem použil externí GPS anténu, a protože byla jasná obloha, GPS čidlo našlo satelity přibližně za 20 sekund po zapnutí. Po krátkém stisknutí tlačítka 1 (viz schéma na obrázku 8) se spustil záznam trasy, GPS čidlo indikovalo status „FIX“ a já jsem započal jízdu, plánovanou přibližně na přibližně 15 minut.

Zařízení/modul/součástka	**Elektrický proud [mA]**
Raspberry Pi samotné	395
Raspberry Pi + kamera (video záznam)	665
Raspberry Pi + GPS	425
Raspberry Pi + ADXL345	nezaznamenáno
Raspberry Pi + kamera + GPS + ADXL345	745

Tabulka 11: Spotřeba elektrického proudu

Během jízdy jsem průběžně sledoval, zda kamera pořizuje záznam a zda GPS čidlo hlásí status „FIX“. Pokud dioda na kameře svítí, pak právě dochází k záznamu videa a pokud se na GPS čidle jednou za 15 sekund rozsvítí dioda, pak GPS má signál a přijímá data - čili vše je v pořádku. Po skončení testovací jízdy jsem krátce stiskl tlačítko 1 (viz obrázek 8), čímž jsem ukončil nahrávání a poté jsem krátce stiskl tlačítko 3, což způsobilo ukončení všech procesů a bylo možno černou skříňku odpojit od zdroje napájení.

Výhled z vozidla, který byl získán z kamery je uveden na obrázku 18 (ponecháno výchozí rozlišení).

Obrázek 18: Výhled z vozidla

Systém pro sběr dat vygeneroval během jízdy tři smyčky, což bylo předpokládáno, protože jsem ponechal nastaveno trvání smyčky 5 minut. Každá smyčka se skládala ze tří souborů - videozáznam ve formátu *h264*, hlasový záznam ve formátu *ogg* a záznam dat z připojených čidel v mém formátu *trackdata*. Ukázku těchto dat je možno vidět v následujícím výpisu trackdata souboru.

```
(lp0
S'interval=1'
p1
aS'speed_units=kmph'
p2
aS'0.0;0.0;0;0.0;0;0000-00-00T00:00:00.000Z;0.0;0.044;1.04'
 ...
p165
aS'18.1622667;49.841515;278.6;39.9664;10;2015-04-23T12:13:21.000Z;0.028;0.128;0.924'
p166
aS'18.1623767;49.84152;278.0;39.1888;10;2015-04-23T12:13:22.000Z;0.148;0.052;0.924'
p167
aS'18.1624867;49.8415217;277.3;38.4292;10;2015-04-23T12:13:23.000Z;-0.004;0.048;1.136'
p168
aS'18.1625883;49.84152;276.3;38.0224;10;2015-04-23T12:13:24.000Z;0.232;0.036;0.96'
p169
aS'18.16269;49.8415133;275.3;38.0944;10;2015-04-23T12:13:25.000Z;0.268;0.052;0.928'
p170
aS'18.162785;49.8415067;274.2;34.408;10;2015-04-23T12:13:26.000Z;0.352;0.084;1.016'
p171
aS'18.16286;49.84149833;273.0;27.4096;10;2015-04-23T12:13:27.000Z;0.168;0.112;0.972'
p172
aS'18.1629317;49.841485;270.2;16.462;10;2015-04-23T12:13:29.000Z;0.076;0.016;0.876'
p173
aS'18.1629317;49.841485;270.2;16.462;11;2015-04-23T12:13:29.000Z;0.156;0.016;1.012'
p174
aS'18.162975;49.8414716;267.1;16.1308;11;2015-04-23T12:13:32.000Z;-0.088;-0.004;0.976'
p175
aS'18.1630033;49.8414767;266.3;10.0188;11;2015-04-23T12:13:33.000Z;0.032;0.092;0.996'
 ...
```

Výpis 6: Ukázka výpisu ze souboru trackdata

Po ukončení testovací jízdy jsem otestoval synchronizaci. Prvně jsem na svém notebooku spustil socketový server pomocí démona *syncdataserverd*. Poté jsem bezdrátově připojil černou skříňku do domácí sítě pomocí Wi-Fi a dlouhým stiskem stiskem tlačítka 1 nastartoval synchronizaci. Došlo k navázání spojení a Raspberry Pi se serveru prokázalo svou identitou - poté započal přenos dat. Bylo přenášeno přibližně 350 MB rychlostí asi 1 Mbit/s a celý proces synchronizace trval cca 7 minut.

Celá testovací jízda proběhla bez nejmenších komplikací. Nebylo potřeba restartovat GPS démona *gpsd* ani nahrávacího démona *trackd*. Však vždy a všude je místo pro vylepšení - co se týče samotného zpracování prototypu černé skříňky, už jen fakt, že černá skříňka byla v papírové krabici poměrně velkých rozměrů, není nejlepší. Zde bych tedy navrhl vylepšení v podobě plastové či hliníkové krabičky a zapojení čidel bez použití nepájivého pole.

Systém byl během sběru dat stabilní - takto soudím, protože jsem při analýze pořízených dat nenalezl žádnou chybu.

5 Závěr

Cílem této diplomové práce bylo navrhnout záznamovou kameru do automobilu, která by měla být použitelná v reálném provozu. S touto kamerou by měl souviset také systém pro prezentaci získaných dat a to vše s využitím mikropočítače Raspberry Pi a různých modulů.

V této práci byla probrána teoretická stránka tématu, která zahrnovala zmapování trhu s podobnými komerčními produkty. Bylo vybráno šest produktů různých cenových kategorií a zohlednil jsem také funkcionalitu, kterou jednotlivé produkty za danou cenu nabízejí.

Dále byla v této práci popsána implementace, na kterou lze nahlížet ze dvou pohledů. Byl popsán hardwarový pohled, kde jsem v úvodu shrnul použité součástky a moduly a uvedl schémata připojení čidel k Raspberry Pi - to vše jsem podrobně popsal. Softwarový pohled jsem rozdělil do částí získávání, synchronizace a prezentace pořízených dat.

Následně byla popsána instalace operačního systému Raspbian na mikropočítač Raspberry Pi. Samozřejmě jsem popsal také samotnou instalaci vytvořeného systému, který jsem nazval „dashcam“ - tato část sestává ze dvou podčástí, a sice instalace systému dashcam na Raspberry Pi a na počítač.

V poslední části práce jsem se věnoval testování vytvořeného systému. Provedl jsem zkušební jízdu a otestoval sběr dat v provozu - s pořízenými daty jsem byl velmi spokojen.

Na tomto místě bych rád porovnal vytvořený systém s komerčními produkty a shrnul přínos práce. Většina komerčních produktů disponuje displejem - systém dashcam displej nemá. Pokud zohledním výše zmíněných 6 komerčních produktů, cenou se systém dashcam pohybuje mírně nad průměrem. Velkou výhodu mého řešení vidím v možnosti jeho rozšíření - pro stávající HW lze doprogramovat různé další funkce. Jak jsem uvedl v úvodní kapitole této diplomové práce, možností rozšíření je spousta a některé z nich jsem popsal v kapitole 4.3. Přínos této diplomové práce vidím v tom, že jsou v ní uvedeny a dobře popsány kroky potřebné k vytvoření systému záznamové kamery pro automobil.

V jednotlivých částech zpracování práce se mi podařilo splnit všechny body zadání. Pořízení dat není obtížná, ale ani triviální záležitost - největší problém při návrhu systému pro mě bylo vymyslet způsob, jak pořízená data prezentovat. Prezentace video (respektive audio) záznamů je velmi jednoduchá, ale analyzovat data z GPS, provést výpočet statistik jízd (respektive zaznamenaných úseků) a pak tyto statistiky prezentovat v ucelené formě pro mě nebylo jednoduché. Avšak podařilo se mi navrhnout způsob prezentace dat, který poskytuje očekávané výsledky.

6 Reference

[1] RICHARDSON, Matt a Shawn P. WALLACE. *Getting started with Raspberry Pi*. 1. vyd. Sebastopol, California: O'Reilly Media, 2012, 161 s. ISBN 14-493-4421-6.

[2] UPTON, Eben. *Raspberry Pi: uživatelská příručka*. 1. vyd. Brno: Computer Press, 2013, 232 s. ISBN 978-80-251-4116-8.

[3] *Oficiální stránky projektu Raspberry Pi*. [online]. [cit. 2015-01-02]. Dostupné z: http://www.raspberrypi.org/

[4] *Raspberry Pi datasheet*. [online]. [cit. 2015-03-02]. Dostupné z: http://www.element14.com/community/docs/DOC-65470/l/raspberry-pi-technical-data-sheet

[5] *Raspberry Pi na en.wikipedia.org*. [online]. [cit. 2015-03-02]. Dostupné z: http://en.wikipedia.org/wiki/Raspberry_Pi

[6] *NMEA data*. [online]. [cit. 2015-04-20]. Dostupné z: http://www.agt.bme.hu/tantargyak/bsc/bmeeoafav49/NMEAdescription_gy_12.pdf

[7] *Dokumentace Raspberry Pi kamery*. [online]. [cit. 2015-04-24]. Dostupné z: http://www.farnell.com/datasheets/1730389.pdf

[8] *Komunikace Pythonu s OBD2*. [online]. [cit. 2015-02-02]. Dostupné z: http://blog.brianhemeryck.me/how-to-interface-with-your-cars-ecu-through-obd2-and-python/

[9] *Automatické rozpoznání registračních značek*. [online]. [cit. 2015-02-02]. Dostupné z: http://en.wikipedia.org/wiki/Automatic_number_plate_recognition

[10] VAINGAST, Shai.*Beginning Python visualization: crafting visual transformation scripts*. Berkeley, CA: Apress, c2009, xx, 363 p. ISBN 9781430218449.

[11] *Adafruit GPS*. [online]. [cit. 2015-02-02]. Dostupné z: https://www.adafruit.com/products/746

[12] *Akcelerometr ADXL343*. [online]. [cit. 2015-02-02]. Dostupné z: http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf

[13] *Projekt kamera pro Raspberry Pi*. [online]. [cit. 2015-02-02]. Dostupné z: http://pidashcam.blogspot.nl/

[14] *VLC Python bindings*. [online]. [cit. 2015-04-26]. Dostupné z: https://wiki.videolan.org/Python_bindings

[15] *Dokumentace knihovny PyQT4*. [online]. [cit. 2015-04-24]. Dostupné z: http://pyqt.sourceforge.net/Docs/PyQt4/classes.html

[16] *Distribuce Raspbian*. [online]. [cit. 2015-05-20]. Dostupné z:
http://www.raspbian.org/

[17] *Oficiální stránky projektu QT Designer*. [online]. [cit. 2015-04-24]. Dostupné z:
http://doc.qt.io/qt-4.8/designer-manual.html

[18] *Kamera GoClever DVR LITE*. [online]. [cit. 2015-01-02]. Dostupné z:
http://40veci.cz/auto-moto/37-25-lcd-hd-autokamera-cerna-skrinka-do-auta.html

[19] *Kamera iGET GUARD F80HD*. [online]. [cit. 2015-01-02]. Dostupné z:
http://www.expert.cz/iget-f80hd

[20] *Kamera TrueCam A3*. [online]. [cit. 2015-01-02]. Dostupné z:
http://www.kameryprosport.cz/truecam-a3

[21] *Kamera CEL-TEC E07*. [online]. [cit. 2015-01-02]. Dostupné z:
http://www.kameryprosport.cz/palubni-kamera-cel-tec-e07

[22] *Kamera TrueCam A5*. [online]. [cit. 2015-01-02]. Dostupné z:
http://www.kameryprosport.cz/truecam-a5

[23] *Kamera Eltrinex LS400W*. [online]. [cit. 2015-01-02]. Dostupné z:
http://www.carvin.cz/kamery-do-auta/autokamera-eltrinex-ls400w-fullhd

[24] *Obrázek Raspberry Pi*. [online]. [cit. 2015-05-01]. Dostupné z:
http://www.rpelectronics.com/Media/400/raspberry-pib.jpg

[25] *Obrázek GPS*. [online]. [cit. 2015-05-01]. Dostupné z:
https://www.kiwi-electronics.nl/image/cache/data/products/componenten/kits/ADA-ULT-GPS-V3-1-800x533.jpg

[26] *Obrázek modulu kamery pro Raspberry Pi*. [online]. [cit. 2015-05-01]. Dostupné z:
http://cdn-reichelt.de/bilder/web/xxl_ws/A300/RASPBERRY_CAM_03.png

[27] *Obrázek akcelerometru ADXL345*. [online]. [cit. 2015-05-01]. Dostupné z:
https://d1dr2mxwsd2nqe.cloudfront.net/media/catalog/product/cache/1/image/9df78eab33525d08d6e5fb8d27136e95/0/9/09836-_01c.jpg

[28] *Obrázek LCD TFT displeje*. [online]. [cit. 2015-05-01]. Dostupné z:
http://i.ebayimg.com/00/s/NTEwWDUxMA==/z/TnsAAOSwQItT7VF5/$_1.JPG?set_id=880000500F

[29] *Použité ikony*. [online]. [cit. 2015-05-05]. Dostupné z:
http://iconsetc.com/

A Obsah CD

Součástí diplomové práce je CD.

Adresářová struktura přiloženého CD:

```
/
├── Dashcam
├── Documents
│   ├── Figures
│   ├── diploma.cls
│   ├── sRGBIEC1966-2.1.icm
│   ├── ruz0024.tex
│   └── ruz0024.pdf
└── Test_data
```

B Inicializační skript - démon *trackd*

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:          trackd
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Data gathering service
# Description:       This service logs data from the sensors and saves them in text database
### END INIT INFO

DIR=/home/pi/dashcam
DAEMON=/usr/bin/python
DAEMON_OPTS=$DIR/modules/button_manager.py
DAEMON_NAME=trackd
DAEMON_USER=root
PIDFILE=$DIR/scripts/$DAEMON_NAME.pid

. /lib/lsb/init-functions

do\_start () {
    log_daemon_msg "Starting system $DAEMON_NAME daemon"
    start-stop-daemon --start --background --pidfile $PIDFILE --make-pidfile --user
        $DAEMON_USER --chuid $DAEMON_USER --startas $DAEMON --
        $DAEMON_OPTS
    log_end_msg $?
}

do_stop () {
    log_daemon_msg "Stopping system $DAEMON_NAME daemon"
    start-stop-daemon --stop --pidfile $PIDFILE --retry 10
    log_end_msg $?
}

case "$1" in
    start|stop)
        do_${1}
        ;;
    restart|reload|force-reload)
        do_stop
        do_start
        ;;
    status)
        status_of_proc "$DAEMON1_NAME" "$DAEMON" && exit 0 || exit $?
        ;;
    *)
        echo "Usage: /etc/init.d/$DAEMON_NAME {start|stop|restart|status}"
        exit 1
        ;;
esac
```